**同济大学计算机系**

**操作系统实验报告**

****

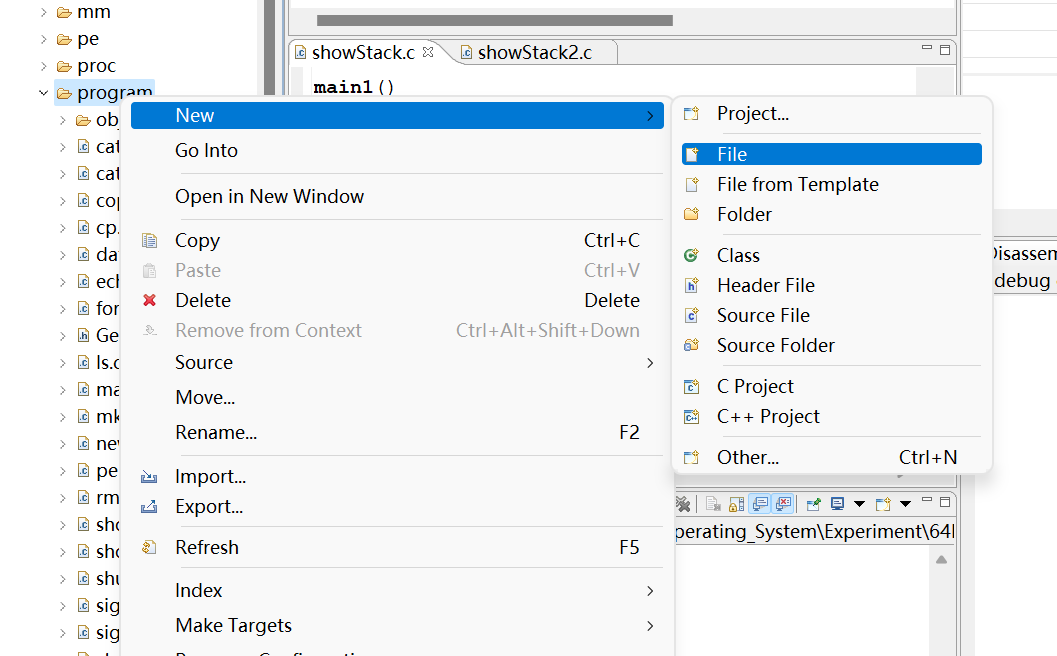
**学 号 2152809**

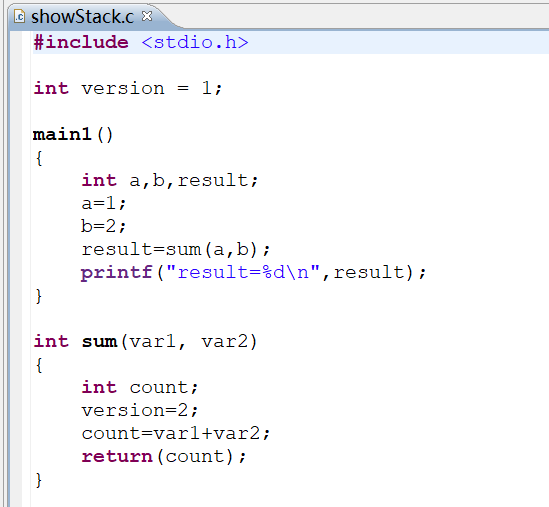
**姓 名 曾崇然**

**专 业 计算机科学与技术**

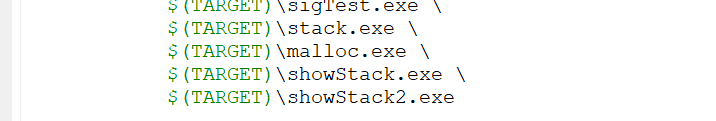
**授课老师 方钰老师**

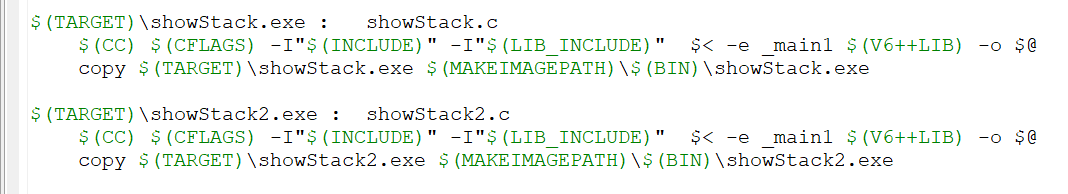
1. **UNIX V6++自定义程序的添加、编译、连接和运行**
2. **自定义程序的添加**
3. 文件的添加与编写：



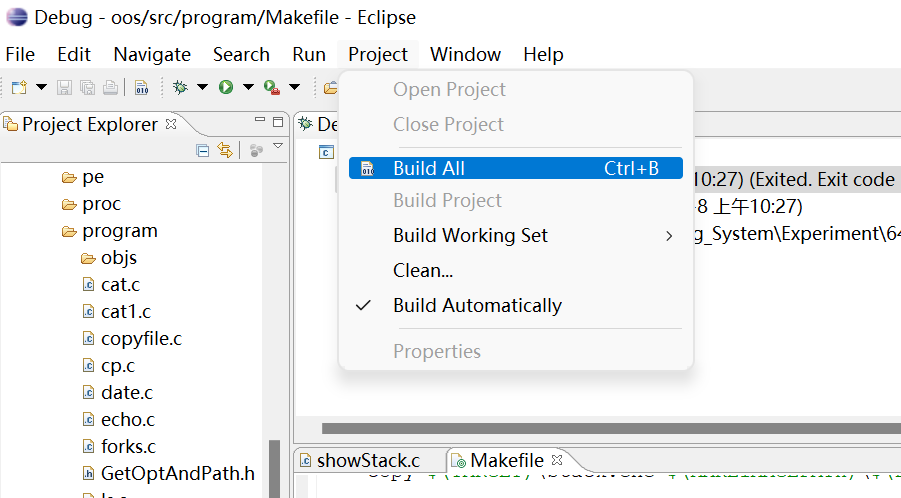


1. 修改配置文件：

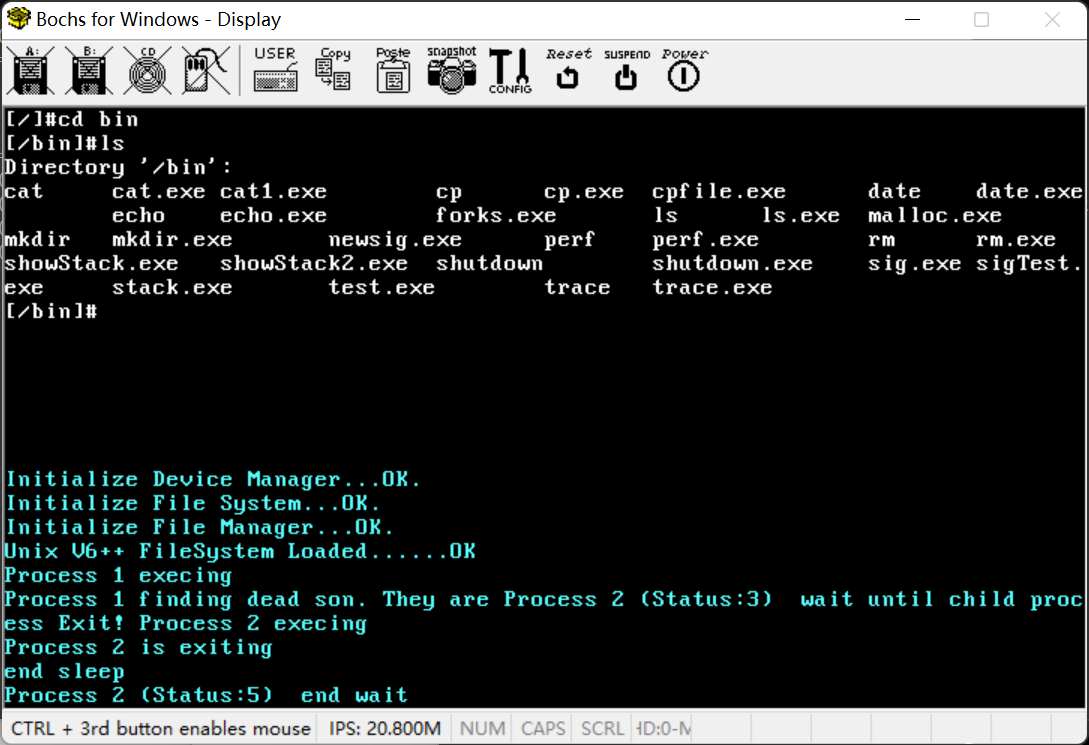




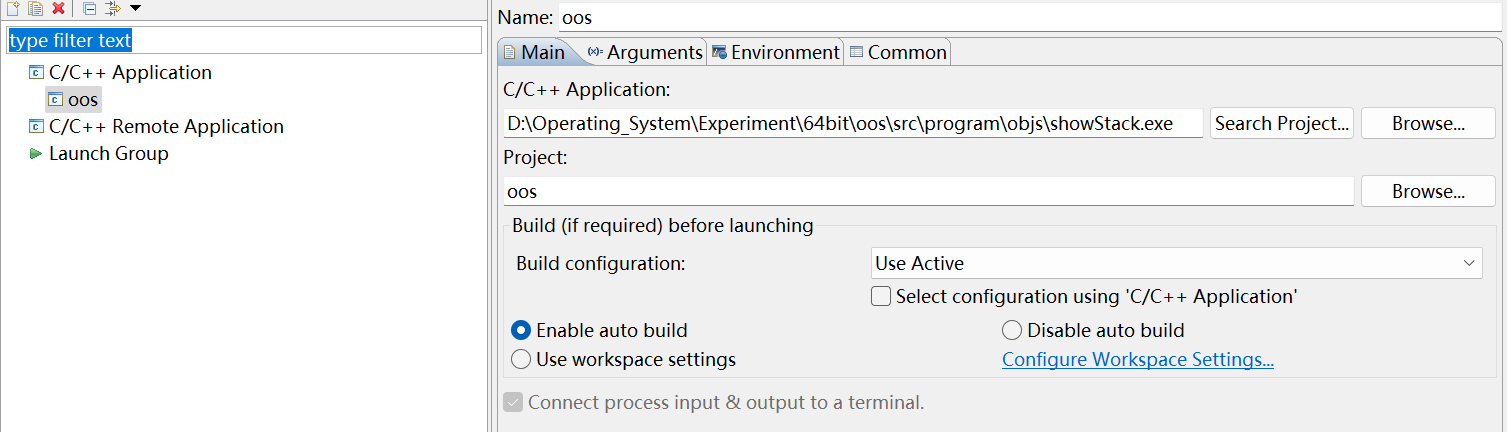
1. **程序的编译**
2. 进行编译：



1. 验证执行文件是否生成：

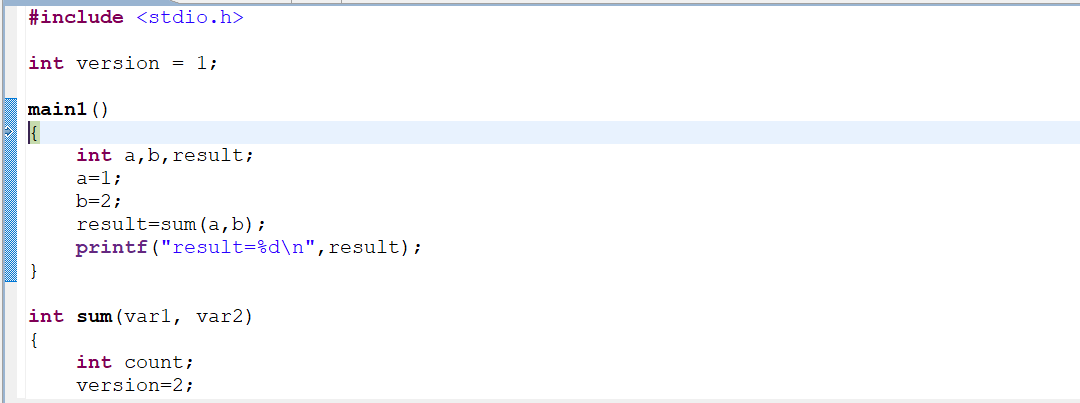


1. **程序的调试和运行**
2. 设置调试开始的起点

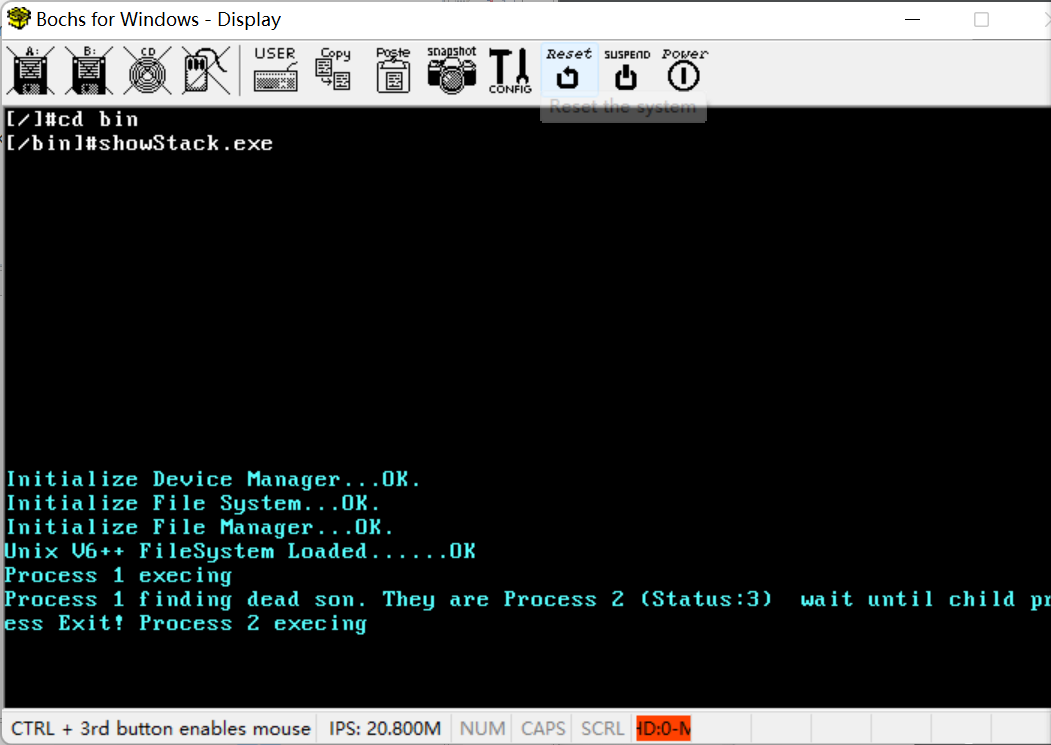


1. 开始调试：

此处停在入口处

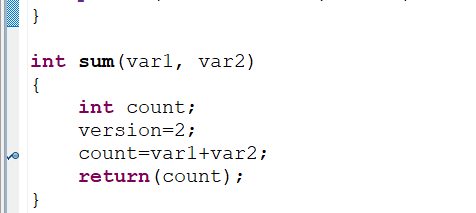


输入命令开始执行和调试

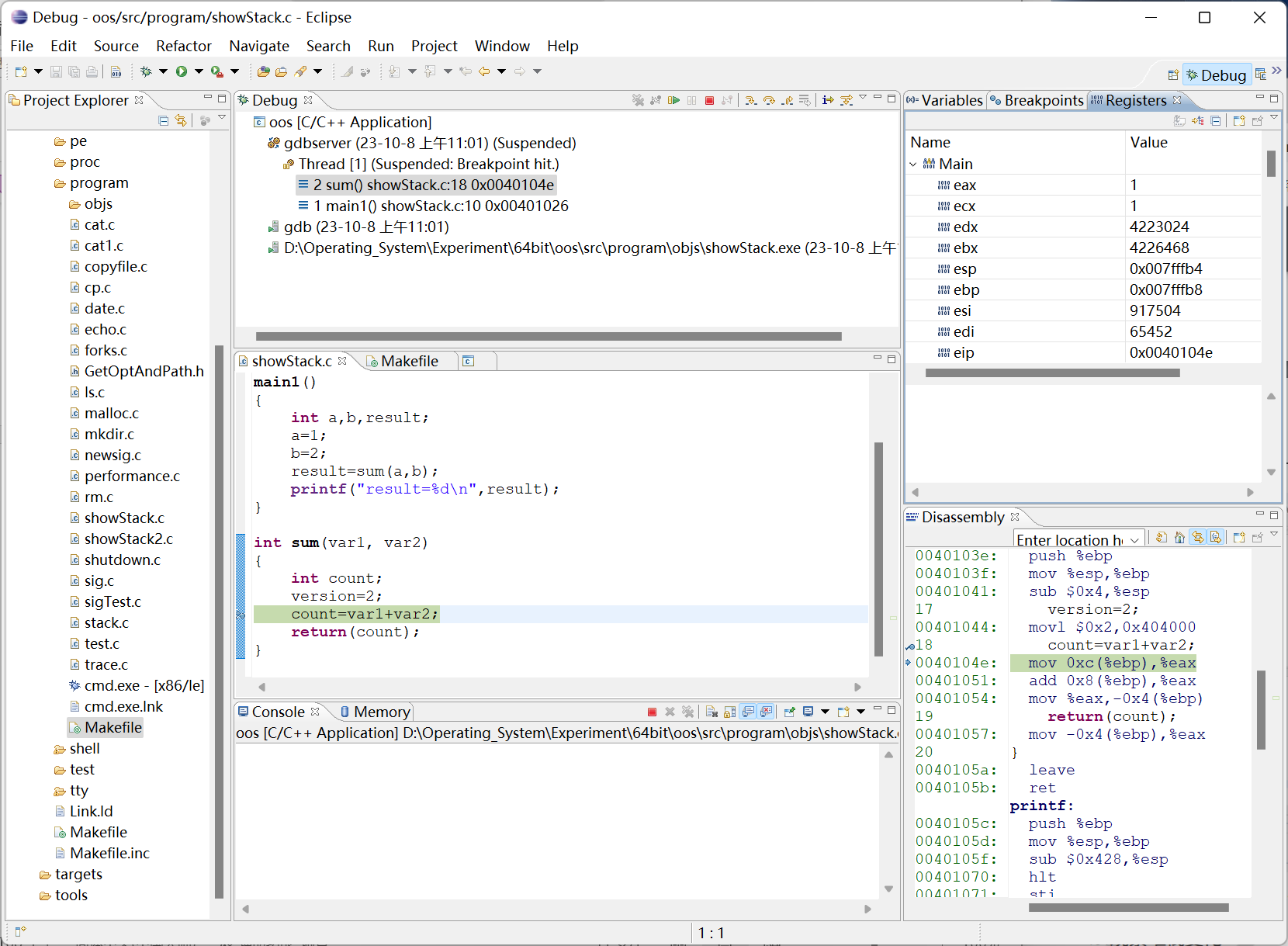


1. 调试运行：

设置断点：

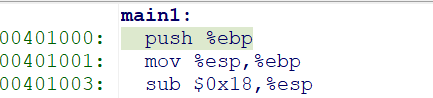


观察寄存器，内存，汇编代码等：

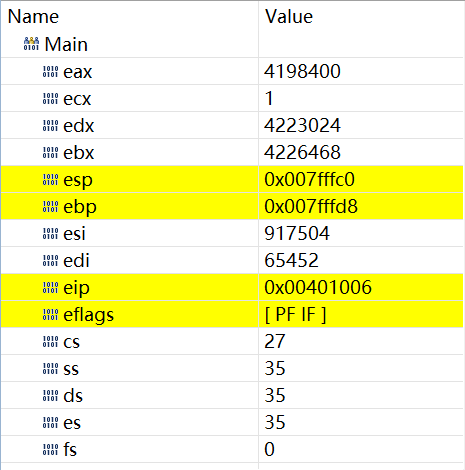


1. **复现并观察核心栈的变化**
2. **存前一栈帧的ebp，修改ebp指向当前栈帧，esp上移**

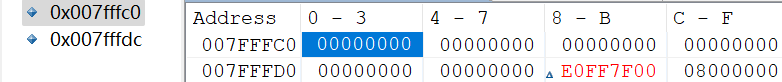
汇编代码：



寄存器值：

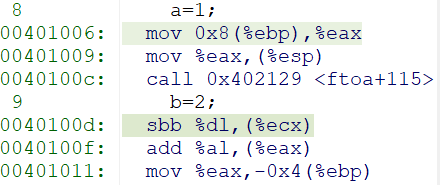


内存单元的观察：观察可知为小端存储，在ebp指向的0x007fffd8单元存储着上一栈帧基址007FFFE0，0x007fffdc存储着main的返回地址00000008，并空出了局部变量和参数的值。

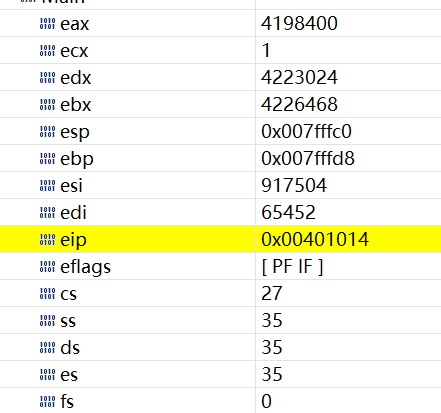


1. **将main的局部变量送入栈中**

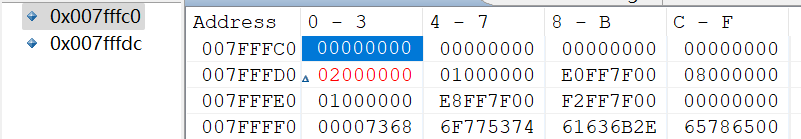
汇编代码：



寄存器值：

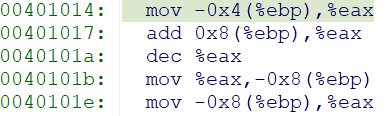


内存单元的观察：可以看到值2和1已经被放入栈中

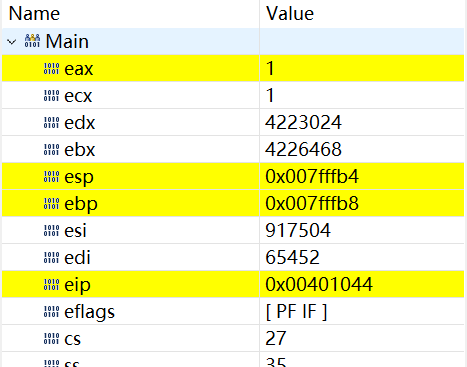


1. **将参数放入栈中**

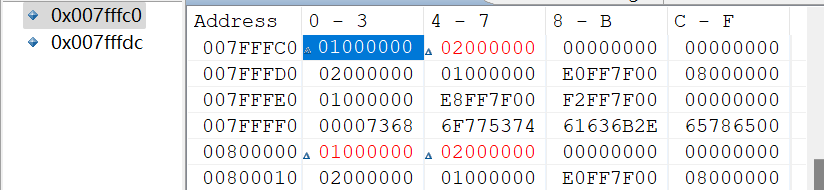
汇编代码：



寄存器值：

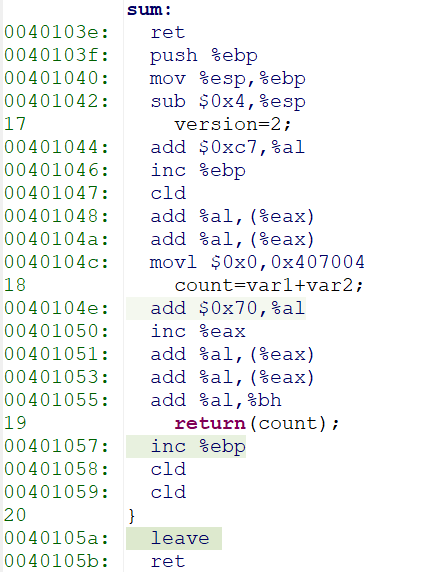


内存单元的观察：参数已经被放入栈中

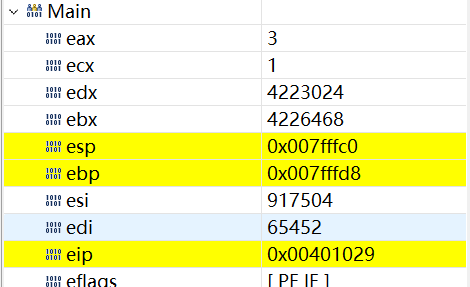


1. **调用sum函数并返回**

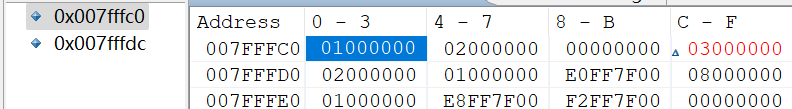
汇编代码：



寄存器值：eax为3，esp重新变回0x007fffc0（和pdf文档中不同？）



内存单元的观察：可以看到为result预留出来的值变为了3



1. **打印，结果如下**



1. **sum代码分析和堆栈的绘制**
2. 代码分析：

sum:

0040103e: push %ebp //将main函数栈帧的ebp存入当前栈

0040103f: mov %esp,%ebp //修改ebp指向当前栈帧

00401041: sub $0x4, %esp //esp上移1个字，空出sum函数局部变量count的位置

17 version=2;

00401044: movl $0x2,0x404000 //将2送入全局变量version中

18 count=var1+var2;

0040104e: mov 0xc(%ebp), %eax //将参数b送入eax中

00401051: add 0x8(%ebp), %eax //将参数a和eax中的参数b相加送入eax中

00401054: mov %eax,-0x4(%ebp) //将相加的结果送入局部变量result中

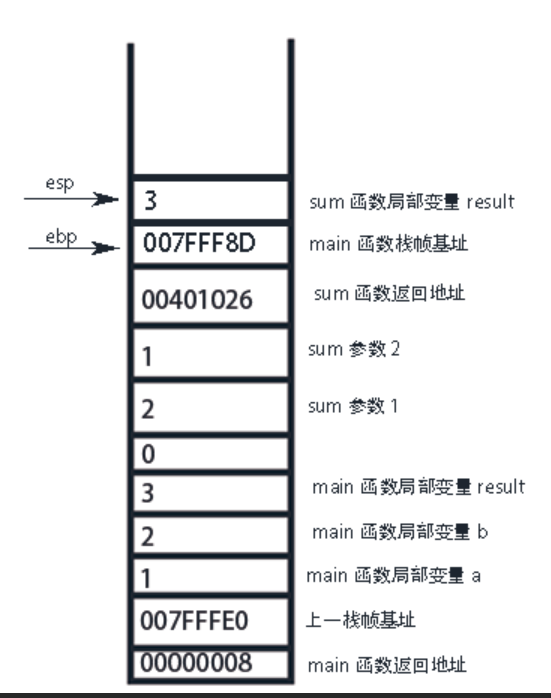
19 return(count);

00401057: mov -0x4(%ebp), %eax //将返回值送入eax中

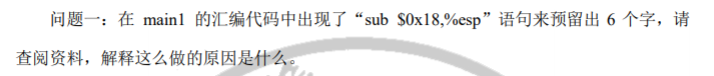
0040105a: leave //撤销当前栈帧

0040105b: ret//返回调用函数前存储的指令位置

1. 完整的栈帧绘制：



1. **问题回答**



* 1. 这是为main函数的局部变量，调用函数的参数分配的预留空间
  2. 注意到该分配的空间大于实际使用的空间，这可能是由于以下的原因：
     1. 出于安全性的考虑，避免局部变量覆盖返回地址等重要数据的可能性
     2. 出于可维护性的考虑，可以轻松添加更多的局部变量而无需调整栈分配的代码
     3. 可能是出于内存单元对齐的要求，多分配了一个字